FLOOD MONITORING SYSTEM USING IOT

AIM:

To build a prototype of an IoT system using Thingspeak cloud and Raspberry Pi and develop a Flood Monitoring System . The aim of the Flood Monitoring System project is to develop a real-time monitoring system using Raspberry Pi and various sensors to detect and alert against potential flooding conditions. The system will measure water levels using an ultrasonic sensor, detect rain using a raindrop sensor, and monitor ambient temperature. It will then use these measurements to trigger alerts and provide data visualization through a mobile/web application.

HARDWARE REQUIREMENTS:

Raspberry Pi , Ultrasonic Sensor, Temperature Sensor , Rain Drop Sensor , Buzzer , LED's.

SOFTWARE REQUIREMENTS:

Raspberyy Pi, Thing Speak Cloud, Python v3.7, Codepen.

THEORY:

A flood monitoring system is a type of environmental monitoring system designed to detect and monitor water levels, rainfall, and other relevant environmental factors in areas prone to flooding. It typically consists of various sensors, data processing units, and communication modules to collect, process, and transmit data in real-time. The main goal of a flood monitoring system is to provide early warning alerts, facilitate decision-making for disaster management authorities, and mitigate the impact of floods on communities and infrastructure.

Flood monitoring system uses IoT devices with Ultrasonic sensor, Temperature sensor, Rain drop sensor to determine the water level in the riverbed and measure the temperature of its surroundings and convert the temperature into a readable output, whenever the predefined threshold limit is reached the buzzer is made on and LED turns red and it sends SMS/ Call/ Email alert to the residents about the flood.

THINGSPEAK:

ThingSpeak is an open-source Internet of Things (IoT) application and API to store and retrieve data from things using the HTTP and MQTT protocol over the Internet or via a Local Area Network. ThingSpeak enables the creation of sensor logging applications, location tracking applications, and a social network of things with status updates.ThingSpeak has integrated support from the numerical computing software MATLAB from MathWorks, allowing ThingSpeak users to analyze and visualize uploaded data using Matlab without requiring a Matlab license from Mathworks. The overview of ThingSpeak is shown in Fig 3.3.

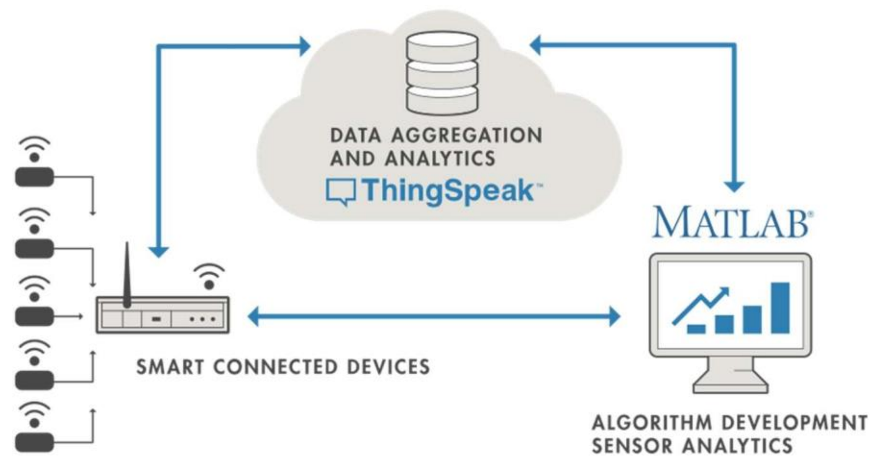

Figure 7.1 Outlook of ThingSpeak

BLOCK DIAGRAM:

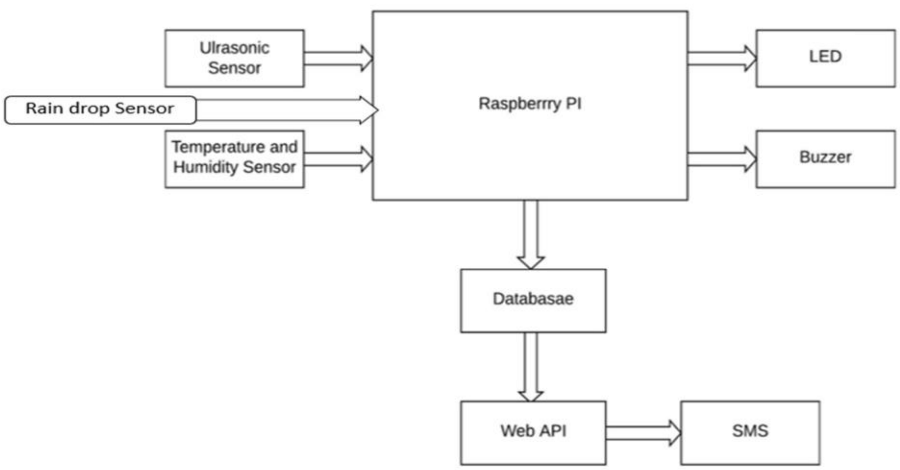

Figure 7.2 Flood Monitoring System Block Diagram

LAYOUT DIAGRAM :

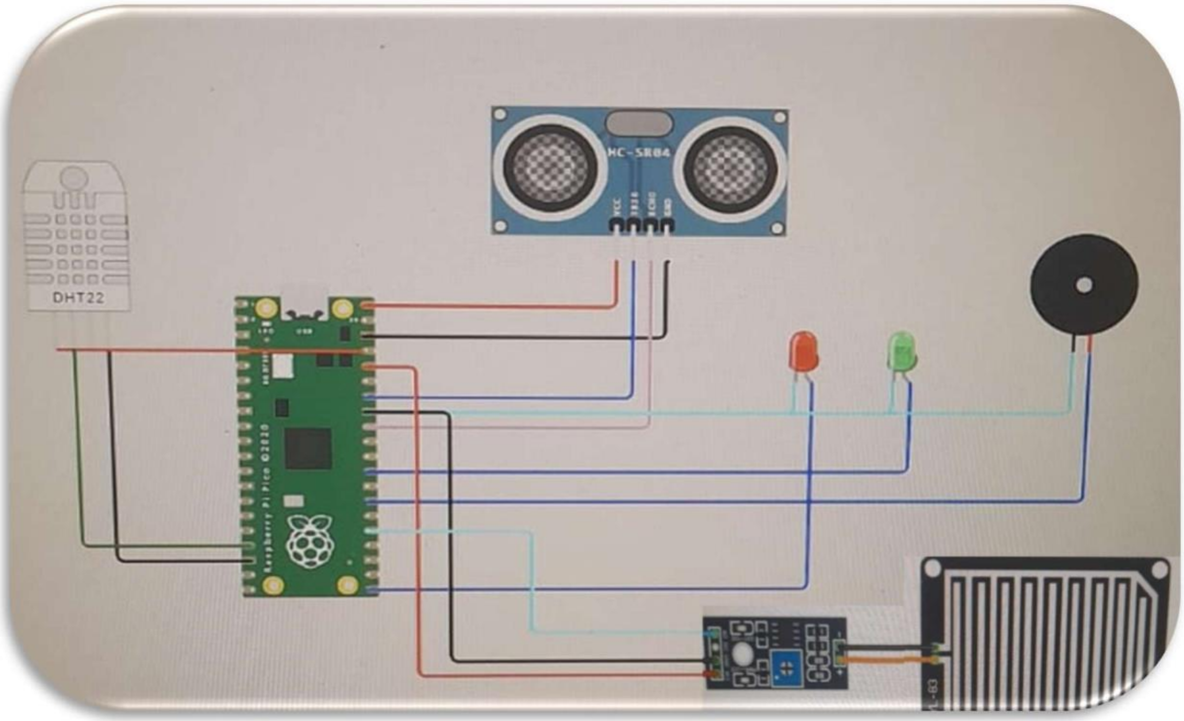

Figure 7.3 Flood Monitoring System Layout Diagram

ALGORITHM:

1. Power up the Raspberry Pi module and Connections are made as per the depicted layout diagram.
2. Raspberry Pi is coded to sense distance , Rain drop sensor status and the temperature and humidity at regular intervals.
3. Create a ThingSpeak channel with suitable fields and generate an API key.
4. Use the API key and generate a base URL to send data to ThingSpeak.
5. The sensed values are uploaded in the respective fields of the ThingSpeak channel using the URL.
6. ThingSpeak takes up to 15 seconds for a value to get uploaded, and then it is visualized using a plot in the cloud.
7. Develop and Web app using Flutter to visualize the project overview and sensor data and weather description .

FLOWCHART:

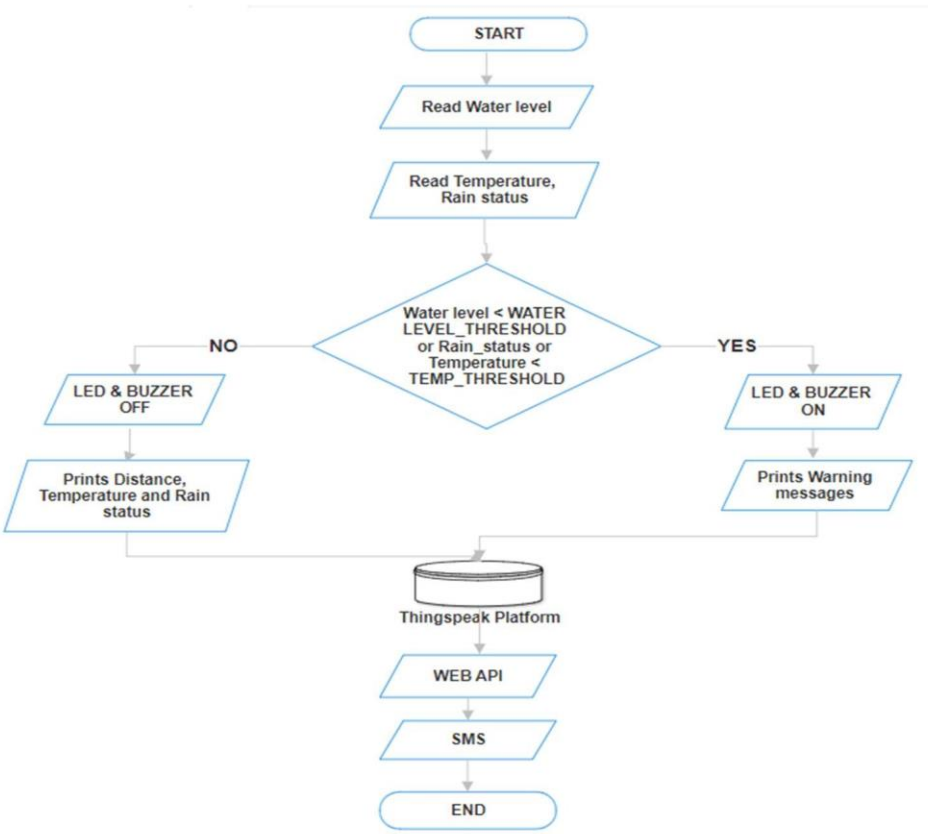

Figure 7.4 Flow Chart for Flood Monitoring System

Raspberry Pi CODE:

```
import RPi.GPIO as GPIO
import time
import Adafruit_DHT
from twilio.rest import Client
import requests
```

```
# ThingSpeak parameters
THINGSPEAK_API_KEY = 'P103E66K4HSTMJMD'
THINGSPEAK_URL = 'https://api.thingspeak.com/update?api_key=P103E66K4HSTMJMD'

# Twilio parameters
TWILIO_ACCOUNT_SID = 'AC6fba5c416c197d9a3ecf1b79baf2c2b0'
TWILIO_AUTH_TOKEN = '8b665b2ecca4a5283de080dd8995f7e8'
TWILIO_PHONE_NUMBER = '+12058431673'
RECIPIENT_PHONE_NUMBER = '+916369587175'

# GPIO pins
ULTRASONIC_TRIG = 21
ULTRASONIC_ECHO = 20
RAIN_SENSOR = 2
RED_LED = 14
GREEN_LED = 15
BUZZER = 18
DHT_PIN = 16
DHT_SENSOR = Adafruit_DHT.DHT11

# Threshold values
ULTRASONIC_THRESHOLD = 3 # Threshold value in centimeters
TEMP_ALERT_THRESHOLD = 27  # Temperature alert threshold in degrees Celsius

# Set GPIO mode
GPIO.setmode(GPIO.BCM)

# Set up ultrasonic sensor
GPIO.setup(ULTRASONIC_TRIG, GPIO.OUT)
GPIO.setup(ULTRASONIC_ECHO, GPIO.IN)

# Set up raindrop sensor
GPIO.setup(RAIN_SENSOR, GPIO.IN)
```

```
# Set up temperature sensor
GPIO.setup(DHT_PIN, GPIO.IN)

# Set up LEDs and Buzzer
GPIO.setup(RED_LED, GPIO.OUT)
GPIO.setup(GREEN_LED, GPIO.OUT)
GPIO.setup(BUZZER, GPIO.OUT)

# Function to measure distance using ultrasonic sensor
def measure_distance():
    GPIO.output(ULTRASONIC_TRIG, True)
    time.sleep(0.00001)
    GPIO.output(ULTRASONIC_TRIG, False)

    pulse_start = time.time()
    pulse_end = time.time()

    while GPIO.input(ULTRASONIC_ECHO) == 0:
        pulse_start = time.time()

    while GPIO.input(ULTRASONIC_ECHO) == 1:
        pulse_end = time.time()

    pulse_duration = pulse_end - pulse_start
    distance = pulse_duration * 17150
    distance = round(distance, 2)

    return distance

# Function to check raindrop sensor
def is_raining():
    # Invert the reading because the rain sensor is producing opposite results
```

```
    return not GPIO.input(RAIN_SENSOR) == GPIO.HIGH

# Function to measure temperature using DHT11 sensor
def measure_temperature():
    humidity, temperature = Adafruit_DHT.read_retry(DHT_SENSOR, DHT_PIN)
    if humidity is not None and temperature is not None:
        temperature = round(temperature, 2)
    return temperature

# Function to send SMS using Twilio
def send_sms(message):
    client = Client(TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN)
    client.messages.create(
        to=RECIPIENT_PHONE_NUMBER,
        from_=TWILIO_PHONE_NUMBER,
        body=message
    )

# Function to update ThingSpeak channel
def update_thingspeak(ultrasonic_distance, temperature):
    payload = {'api_key': THINGSPEAK_API_KEY, 'field1': ultrasonic_distance, 'field2': temperature}
    try:
        requests.post(THINGSPEAK_URL, params=payload)
    except Exception as e:
        print("Error updating ThingSpeak:", e)

# Main loop
try:
    while True:
        ultrasonic_distance = measure_distance()
        rain_status = is_raining()
        temperature = measure_temperature()
```

```
        print("Ultrasonic Distance:", ultrasonic_distance, "cm")
        print("Temperature:", temperature, "C")

        if ultrasonic_distance < ULTRASONIC_THRESHOLD or rain_status or temperature <
TEMP_ALERT_THRESHOLD:
            GPIO.output(RED_LED, GPIO.HIGH)
            GPIO.output(GREEN_LED, GPIO.LOW)
            GPIO.output(BUZZER, GPIO.HIGH)
            send_sms("Alert: Water level exceeded threshold or Rain detected or Temperature below
{}C!".format(TEMP_ALERT_THRESHOLD))
        else:
            GPIO.output(RED_LED, GPIO.LOW)
            GPIO.output(GREEN_LED, GPIO.HIGH)
            GPIO.output(BUZZER, GPIO.LOW)

        update_thingspeak(ultrasonic_distance, temperature)

        time.sleep(5) # Adjust according to your requirement

except KeyboardInterrupt:
    GPIO.cleanup()
```

RESULT AND DISCUSSIONS:

The connection setup is shown in Figure 7.4. The distance readings, rain status and temperature readings are stored in the Cloud as shown in Figure 7.5 and Figure 7.6 which vary for different cases.

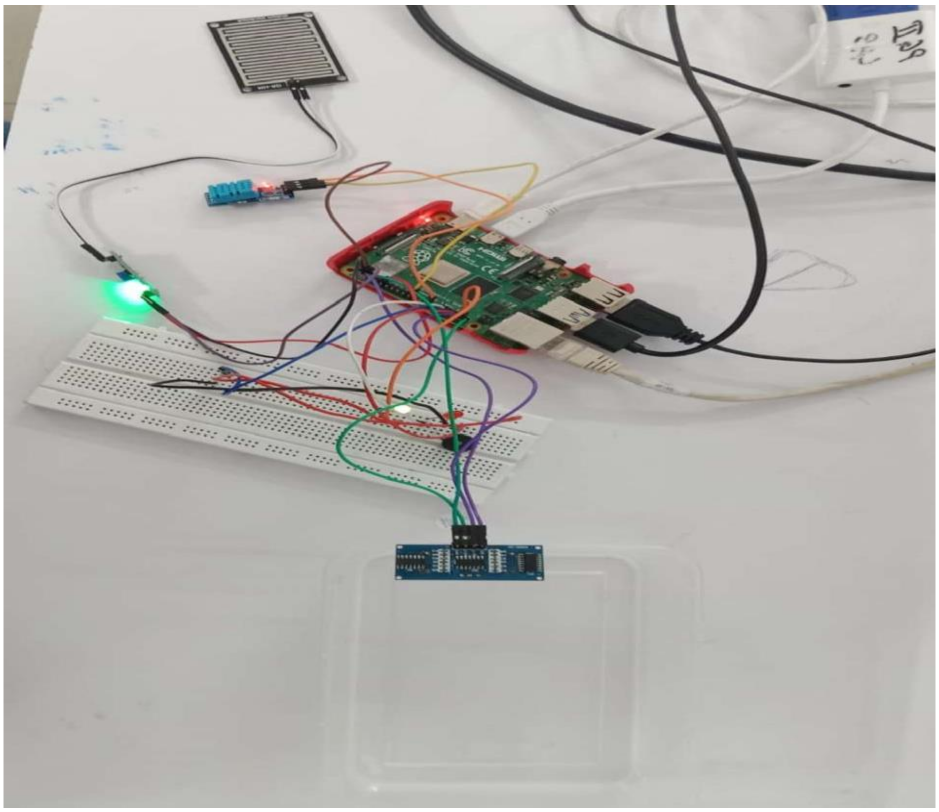

Figure 7.5 Flood Monitoring System connection setup

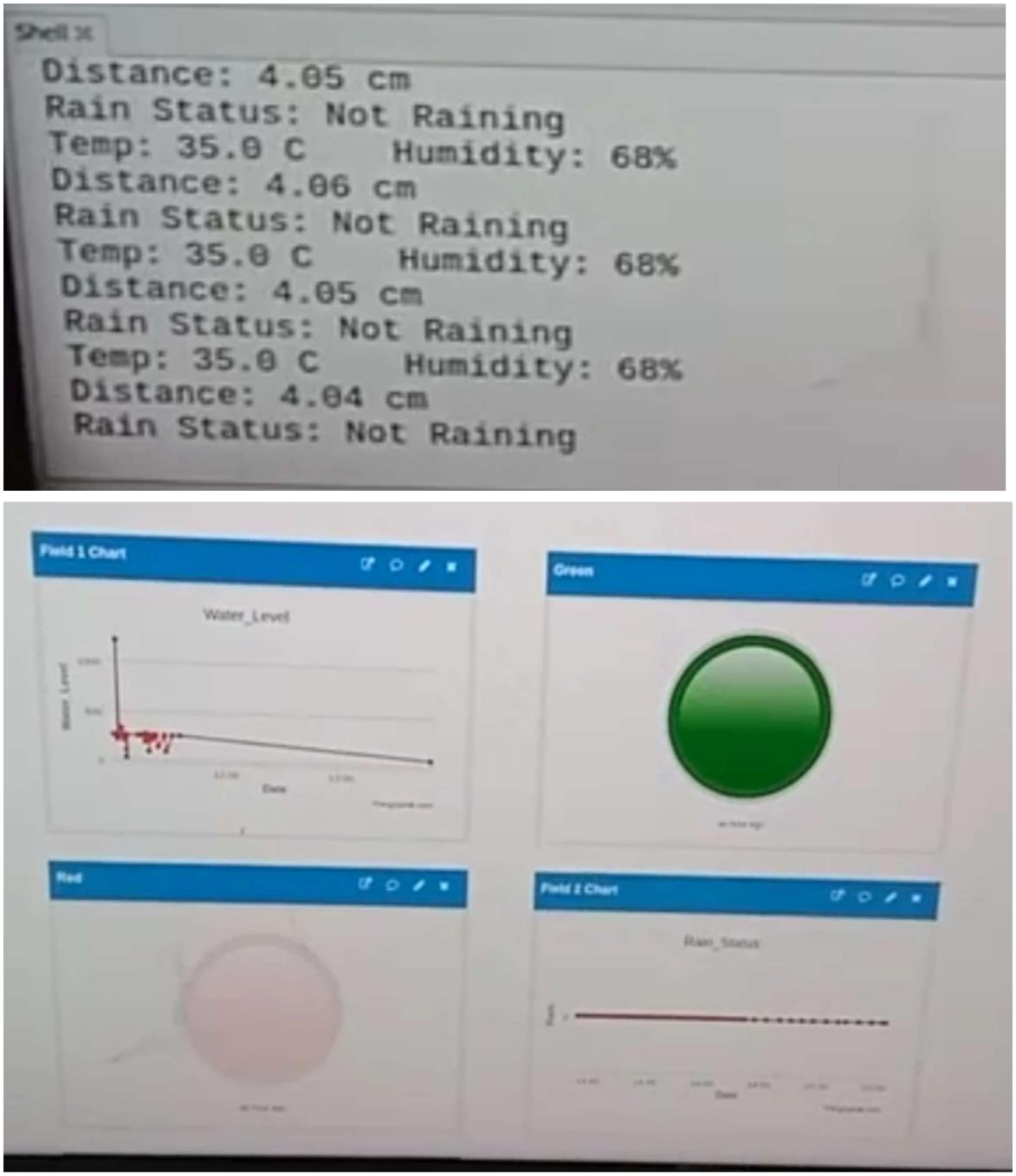

Figure 7.6 Thingspeak channel Output for Idle Status

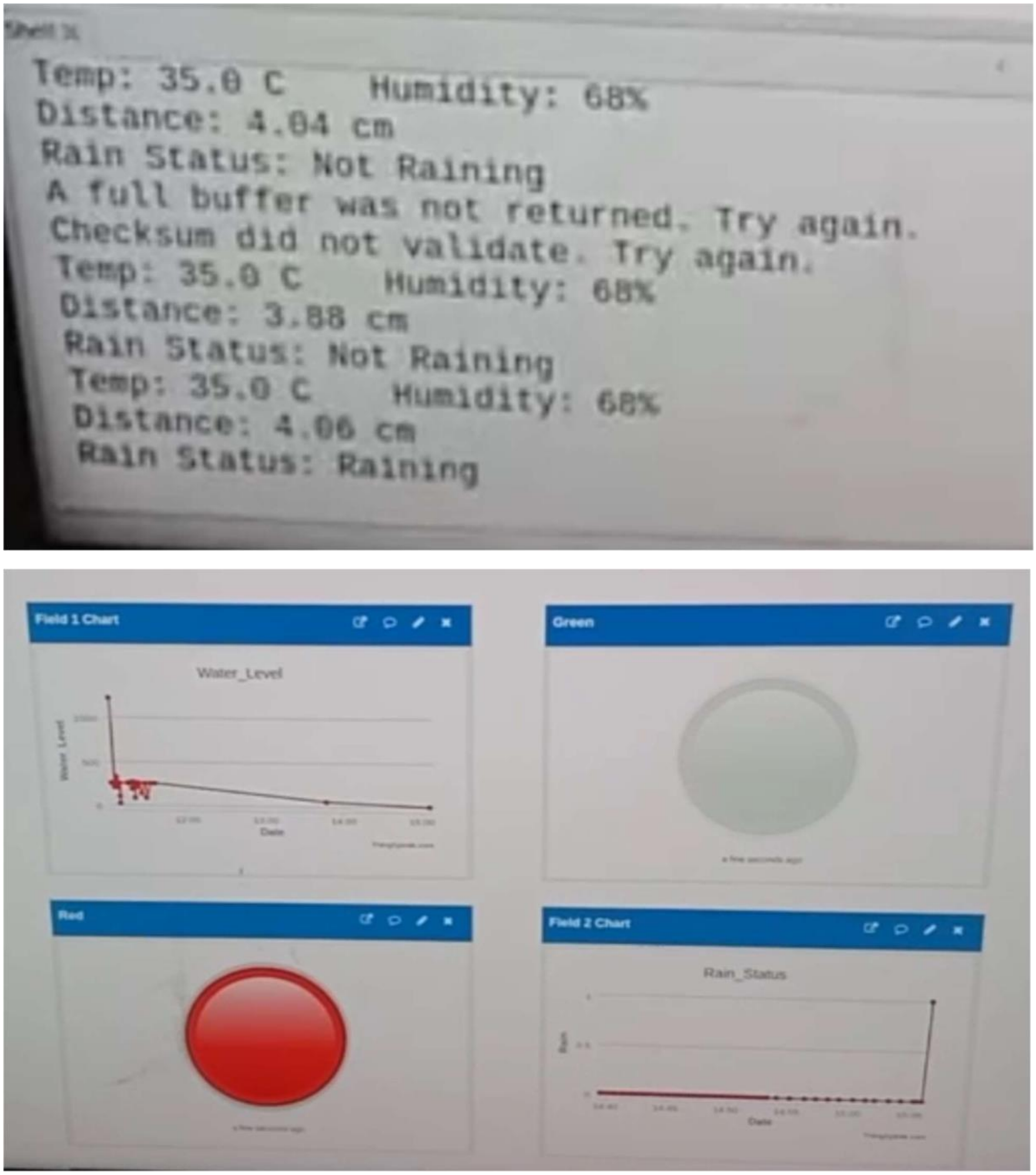

Figure 7.7 ThingSpeak Channel Output for Raining Status

CONCLUSION :

Hence , a prototype of an IoT system using Thingspeak cloud and Raspberry Pi for Flood Monitoring System is developed and the real-time readings have been observed and data has been logged to ThingSpeak. A real-time monitoring system using Raspberry Pi and various sensors to detect and alert against potential flooding conditions is implemented successfully . The system will measure water levels using an ultrasonic sensor, detect rain using a raindrop sensor, and monitor ambient temperature. It will then use these measurements to trigger alerts and provide data visualization through a connected mobile/web application.